BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

IN THE CLAIMS:

Please amend the claims as set forth below:

1. (Original) A storage comprising:

a non-volatile memory storing a first inode corresponding to a first file; and

a block manager configured to copy said first inode to a second inode, wherein said block manager is configured to change said second inode in response to updates to said first file, and wherein said block manager is configured to atomically update said first file in response to a commit of said first file by writing said second inode to said non-volatile memory.

2. (Original) The storage as recited in claim 1 wherein said non-volatile memory stores a journal comprising a list of committed inodes, and wherein said block manager is configured to record said second inode in said journal.

3. (Original) The storage as recited in claim 2 wherein said commit of said first file comprises a commit command received from an external source which updates said first file.

4. (Original) The storage as recited in claim 3 wherein said commit command comprises a file close command.

5. (Original) The storage as recited in claim 3 wherein said commit command comprises an fsync command.

6. (Original) The storage as recited in claim 2 wherein said journal further includes a checkpoint record including a description of an inode file, a block allocation bitmap, and an inode allocation bitmap.

7. (Original) The storage as recited in claim 6 wherein the description comprises inodes for each of said inode file, said block allocation bitmap, and said inode allocation bitmap.

8. (Original) An apparatus comprising:

a computing node configured to perform one or more write commands to a file and a commit command committing the one or more write commands to said file; and

a storage coupled to receive said one or more write commands and said commit command, wherein said storage is configured to copy one or more blocks of said file to a copied one or more blocks, said one or more blocks updated by said one or more write commands, and wherein said storage is configured to update said copied one or more blocks with write data corresponding to said one or more write commands, and wherein said storage is configured to copy a first inode corresponding to said file to a second inode and to update pointers within said second inode corresponding to said one or more blocks to point to said copied one or more blocks, and wherein said storage is configured to atomically update said file by writing said second inode responsive to said commit command, and wherein said first inode is stored in an inode file, and wherein said inode file is identified by a master inode, and wherein said inode file is atomically updated with said second inode by writing said master inode subsequent to said commit command.

9. (Original) The apparatus as recited in claim 6 wherein said commit command comprises a file close command.

10. (Original) The apparatus as recited in claim 6 wherein said commit command comprises an fsync command.

11. (Original) A method comprising:

copying a first inode corresponding to a first file to a second inode;

modifying said second inode in response to one or more changes to said first file; and

atomically updating said first file by establishing said second inode as the inode for said first file.

12. (Original) The method as recited in claim 11 wherein said establishing comprises storing said second inode in a journal stored in a nonvolatile memory.

13. (Original) The method as recited in claim 12 further comprising writing a master inode corresponding to an inode file including said second inode to a checkpoint record in said journal.

14. (Original) The method as recited in claim 13 wherein recovering from a system failure comprises:

scanning said journal to locate a most recent checkpoint record and zero or more inodes subsequent to said most recent checkpoint record within said journal;

copying said master inode from said most recent checkpoint record to a volatile memory; and

updating an inode file corresponding to said master inode with said one or more inodes subsequent to said most recent checkpoint record.

15. (Original) The method as recited in claim 14 wherein said updating said inode file

comprises:

copying one or more blocks of said inode file storing said one or more inodes to a copied one or more blocks; and

updating said master inode in said volatile memory to point to said copied one or more blocks.

16. (Original) The method as recited in claim 11 wherein said block map further comprises a first inode allocation bitmap indicating which inodes within said first inode file are allocated to files, the method further comprising:

copying said first inode allocation bitmap to a second inode allocation bitmap;

modifying said second inode allocation bitmap to reflect one or more inodes allocated to new files; and

establishing a second inode within said block map to said second inode allocation bitmap subsequent to said modifying said second inode bitmap.

17. (Original) The method as recited in claim 16 wherein said block map further comprises a first block allocation bitmap indicating which blocks within a storage including said block map are allocated to files, the method further comprising:

copying said first block allocation bitmap to a second block allocation bitmap;

modifying said second block allocation bitmap to reflect one or more blocks allocated to files; and

establishing a third inode within said block map to said second block allocation bitmap subsequent to said modifying said second block allocation bitmap.

18. (Original) The method as recited in claim 11 wherein said establishing said second inode is performed in response to a commit command.

19. (Original) The method as recited in claim 18 wherein said commit command is a close file command.

20. (Original) The method as recited in claim 18 wherein said commit command is an fsync command.

21. (Original) A storage comprising:

a non-volatile memory storing a first inode corresponding to a first version of a file; and

a block manager configured to copy said first inode to a second inode, wherein said block manager is configured to change said second inode in response to updates to the file, and wherein said block manager is configured to atomically update the file, producing a second version of the file, in response to a commit of the file by writing said second inode to said non-volatile memory.

22. (Original) The storage as recited in claim 21 wherein said non-volatile memory stores a journal comprising a list of committed inodes, and wherein said block manager is configured to record said second inode in said journal.

23. (Original) The storage as recited in claim 22 wherein said commit of the file comprises a commit command received from an external source which updates the file.

24. (Original) The storage as recited in claim 23 wherein said commit command comprises a file close command.

25. (Original) The storage as recited in claim 23 wherein said commit command comprises an fsync command.

26. (Original) The storage as recited in claim 22 wherein said journal further includes a checkpoint record including a description of an inode file, a block allocation bitmap, and an inode allocation bitmap.

27. (Original) The storage as recited in claim 26 wherein the description comprises inodes for each of said inode file, said block allocation bitmap, and said inode allocation bitmap.

28. (Original) A method comprising:

copying a first inode corresponding to a first version of a file to a second inode;

modifying said second inode in response to one or more changes to the file, creating a second version of the file; and

atomically updating the file to the second version by establishing said second inode as the inode for the file.

29. (Original) The method as recited in claim 28 wherein said establishing comprises storing said second inode in a journal stored in a nonvolatile memory.

30. (Original) The method as recited in claim 29 further comprising writing a master inode corresponding to an inode file including said second inode to a checkpoint record in said journal.

31. (Original) The method as recited in claim 30 wherein recovering from a system failure comprises:

scanning said journal to locate a most recent checkpoint record and zero or more inodes subsequent to said most recent checkpoint record within said journal;

copying said master inode from said most recent checkpoint record to a volatile memory; and

updating an inode file corresponding to said master inode with said one or more inodes subsequent to said most recent checkpoint record.

32. (Original) The method as recited in claim 31 wherein said updating said inode file comprises:

copying one or more blocks of said inode file storing said one or more inodes to a copied one or more blocks; and

updating said master inode in said volatile memory to point to said copied one or more blocks.

33. (Original) The method as recited in claim 28 wherein said block map further comprises a first inode allocation bitmap indicating which inodes within said first inode file are allocated to files, the method further comprising:

copying said first inode allocation bitmap to a second inode allocation bitmap;

modifying said second inode allocation bitmap to reflect one or more inodes allocated to new files; and

establishing a second inode within said block map to said second inode allocation bitmap subsequent to said modifying said second inode bitmap.

34. (Original) The method as recited in claim 33 wherein said block map further comprises a first block allocation bitmap indicating which blocks within a storage including said block map are allocated to files, the method further comprising:

copying said first block allocation bitmap to a second block allocation bitmap;

modifying said second block allocation bitmap to reflect one or more blocks allocated to files; and

establishing a third inode within said block map to said second block allocation bitmap subsequent to said modifying said second block allocation bitmap.

35. (Original) The method as recited in claim 28 wherein said establishing said second inode is performed in response to a commit command.

36. (New) A storage comprising:

a non-volatile memory storing a first file; and

a block manager coupled to receive a plurality of write commands and a commit command from an interconnect to which the storage is coupled during use, wherein the plurality of write commands update the first file, and wherein the commit command is defined to commit the updates to the first file, wherein the block manager is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command.

37. (New) The storage as recited in claim 36 wherein the first file in the non-volatile memory is a first version of the first file, and wherein the block manager is configured to create a second version of the first file in response to a first write command of the

plurality of write commands, and wherein the block manager is configured to atomically replace the first version with the second version in response to the commit command.

38. (New) The storage as recited in claim 37 wherein the first version is associated with a first inode, and wherein the second version is created by copying the first inode to a second inode and modifying the second inode, and wherein the atomic update is performed by writing the second inode.

39. (New) The storage as recited in claim 36 wherein the storage is an object-based storage and wherein the plurality of write commands and the commit command address a file object corresponding to the first file.

40. (New) An apparatus comprising:

a computing node configured to generate a plurality of write commands to update a first file and a commit command defined to commit the updates to the first file;

an interconnect to which the computing node is coupled, wherein the computing node is configured to transmit the plurality of write commands and the commit command on the interconnect; and

a storage coupled to the interconnect and configured to store the first file, wherein the storage is configured to atomically update the first file to reflect the plurality of write commands responsive to the commit command.

41. (New) The apparatus as recited in claim 40 wherein the first file on the storage prior to the plurality of write commands is a first version of the first file, and wherein the storage is configured to create a second version of the first file in response to a first write command of the plurality of write commands, and wherein the storage is configured to atomically replace the first version with the second version in response to the commit

command.

42. (New) The storage as recited in claim 41 wherein the first version is associated with a first inode, and wherein the second version is created by copying the first inode to a second inode and modifying the second inode, and wherein the atomic update is performed by writing the second inode.

43. (New) The storage as recited in claim 40 wherein the storage is an object-based storage and wherein the plurality of write commands and the commit command address a file object corresponding to the first file.